A perturbative clustering hyper-heuristic framework for the Danish railway system

M. Pour, Shahrzad; Rasmussen, Kourosh Marjani; Burke, Edmund K.

Publication date:
2015

Document Version
Publisher's PDF, also known as Version of record

A perturbative clustering hyper-heuristic framework for the Danish railway system

Shahrzad M.Pour, PhD student

DTU Management Engineering, Technical University of Denmark, Produktionstorvet, 2800 Kgs. Lyngby, Denmark

Kourosh Marjani Rasmussen, associate professor

DTU Management Engineering, Technical University of Denmark, Produktionstorvet, 2800 Kgs. Lyngby, Denmark

Edmund K. Burke, professor

Queen Mary University of London, Queen's Building, Mile End Road, London E1 4NS, UK

Abstract

A new signaling system in Denmark aims to ensure fast and reliable train operation, imposing very strict time limits on recovery plans. This makes it necessary to rethink the whole maintenance scheduling process. In the largest region of Denmark, the Jutland peninsula, there is a decentralized structure for maintenance planning, where the crew start their duties from different locations rather than starting from a single depot. In this paper, we partition the Jutland problem into subregions before the scheduling phase, according to the tasks and crew locations. To undertake this region splitting, we propose a perturbative clustering hyper-heuristic framework. The framework improves an initial solution by reassigning outliers (those tasks that are far away) to a better cluster choice at each iteration while taking balanced crews workloads into account. The framework introduces five low-level heuristics and employs an adaptive choice function as a robust learning mechanism. The results of adaptive clustering hyper-heuristic are compared with two exact and heuristic assignment algorithms from the literature and

Email addresses: shmp@dtu.dk (Shahrzad M.Pour, PhD student), kmra@dtu.dk (Kourosh Marjani Rasmussen, associate professor), e.burke@qmul.ac.uk (Edmund K. Burke, professor)

with the random hyper-heuristic framework on 12 datasets. In comparison with the exact formulation, the proposed framework could obtain promising results and solved the data instances up to 5000 number of tasks. In comparison with heuristic assignment and the random hyper-heuristic, the framework yielded approximately 11%, 27% and 10%,13% improvement on total distance and the maximum distance availability, respectively. Finally, to assess the closeness of the tasks within each cluster the compactness measure was compared across the three different solutions.

Keywords: Combinatorial Optimization, hyper-heuristic, maintenance scheduling, transportation, partitioning, European Rail Traffic Management System

---

1. Introduction

The introduction of a common European Rail Traffic Management System (ERTMS) [1] throughout Europe aims at enhancing the connectivity between European cities and capitals. The ERTMS is a major European industrial project to allow for faster travel within the EU by creating a unified Europe-wide standard for railway signaling. Since ERTMS is in its early stage of operation, only few research works treated the maintenance issues related to the ERTMS project [2]. Denmark will be the first country in Europe to upgrade its entire signaling system to ERTMS at once. The railway track and signalling system is a complex, highly interdependent system, where the failure of one part, e.g. a train signal, can shut down large parts of the whole system. Given the large investments in tracks and signalling system, Denmark is right now investing approx. 3 billion Euro in the signals[3], correct maintenance becomes even more important. Banedanmark, a state-owned Danish company responsible for maintenance and traffic control of most of the Danish railway network, wishes to develop such a system.

Using completely different hardware in the new system requires different maintenance tasks, consequently very strict time limits and constraints on recovery operations compared to the existing system. To handle the unexpected failures and the breakdowns, there should be at least one crew available at the breakdown location within a certain time limit. According to the industrial partner of the ERTMS project in Denmark, the maintenance plan should help in defining the subregions, in which each crew is working. The workload in each sub-region should be balanced, the regions should be logical, and the geography of the regions should ensure that crew

can come from any place in the region to any other place in the region in short time when needed by corrective maintenance.

The focus of this paper is on partitioning of the maintenance tasks for the Jutland peninsula, the largest region of Denmark. The reason behind this idea is the decentralized structure of the current maintenance planning in Denmark, where the crew start their duties from different locations rather than starting from a single depot. This structure calls for a more accurate assignment of the tasks to the crew to avoid a high total driving distance cost or even finding a feasible plan. In this way, we make each crew responsible for undertaking the tasks in their own subregion.

According to the literature, partitioning is a sub class of clustering problem. Due to various practical application of the clustering concept, the notation of "cluster" in not precisely defined [4]. It embraces various scientific disciplines from mathematics and statistics to biology and genetics [5], [6], [7]. But, the problem is identical for all the fields: forming categories of entities and assigning individuals to the proper groups within it. The most well-known clustering algorithms are hierarchical clustering, partitioning methods, density-based, model-based, grid-based and soft-computing methods [8]. In addition, because the clustering problem is an NP-hard problem [9], heuristics [10] have been broadly used in finding an approximate solution of such problem in a reasonable amount of time, too.

The partitioning problem we deal with in this paper is similar to clustering techniques of [11], [8], [12], all of which use a pre-specified number of clusters. The most well-known approaches are K-means [13] and K-medoids [14]. These approaches create an initial set of k partitions, where parameter k is the number of partitions to construct. They then use an iterative relocation technique that attempts to improve the partitioning by moving objects from one group to another. However, by changing their cluster representatives they cannot be a choice for our partitioning problem.

Since the region splitting is being used before scheduling phase, the whole crew scheduling maintenance problem can be seen as Multi Depot Vehicle Routing Problem[15], where the vehicles start and end their routes at different depots. The MDVRP in general can be seen as the clustering problem in the sense that the result is a set of vehicle schedules clustered by depot. This interpretation proposes a class of approaches that clusters customers as a prior phase to scheduling the vehicles over each cluster[16]. Accordingly, in cluster-first, route-second approaches, the clustering part of the approaches is usually solved by assignment algorithms [17]. In the literature, there are mainly four different classes of assignment algorithms to deal with MDVRP problems. The main class of these approaches are assignment through urgen-

cies which define a precedence relationship between customers to determine the order in which customers are allotted to depot. The heuristics of Parallel assignment [18], the Simplified assignment [16] and the Sweep assignment belong to this category and are only different in the precedence order they use to assign the customers. Another category of assignment approaches are assigning the customers to depots through formulation of the Assignment Problem as a sub class of Transportation Problem. This gives an assignment of the customers to depot through an exact approach.

Presently, many scholars have paid attention to use a hyper-heuristic framework for solving combinatorial optimization problems, [19]. The term hyper-heuristic was coined in the early 2000s [20] to refer to the idea of heuristics to choose heuristics. The latest classification of hyper-heuristics is proposed in [21] according to the nature of the heuristic search space, and the source of feedback during learning. Considering the nature of the heuristics search space, low-levels could be either constructive or perturbative. Several approaches have recently been proposed to generate efficient construction heuristics in different domains [22], [23], [24]. Hyper-heuristics based on perturbation have been applied to different domain such as personnel scheduling, timetabling and vehicle routing problems [25], [26]. However, quite few hyper-heuristics has been studied for the clustering problem in the literature. A hyper-heuristic clustering algorithm based on two diversification and intensification detector presented in [27]. Another hyper-heuristic called HHWDC for web document clustering is presented in [28]. In software fields, a fast multi-objective hyper-heuristic Genetic Algorithm (MHypGA) for the solution of Multi-objective Software Module Clustering Problem has been represented in [29].

This paper is organized into six sections. In the next section, we present the mathematical model of the partitioning problem. Section three explains the proposed clustering hyper-heuristic framework used as region splitter for the Danish railway systems. Section four describes the dataset and section five presents the experimental result and discussion of the proposed framework. Finally, this paper encloses with a conclusion in Section six.

2. Mathematical model

The mathematical model of the partitioning problem we are dealing with in this paper is mainly an initial task assignment to the crew/clusters before scheduling phase. The objective function (1) is multi criteria and the first term in the objective function minimizes the total travel time from a crew to the assigned tasks for that crew. The second term aims at minimizing the

maximum distance of the pair tasks within each cluster. In addition, fair distribution of the tasks among the crew is considered as a third criteria in this phase to reduce the complexity of the problem in routing and scheduling phase. w is the upper bound for work load balancing mismatches across different clusters as described by constraint (4). We consider the balancing in the task duration as fair distribution of the tasks factor in our model.

Constraint (2) imposes that each task should be assigned to only one crew. Together with second term ψ in the objective function, constraint (3) minimize the maximum distance among pair tasks within each cluster. This reflects the definition of the diameter of a cluster as the maximum distance between any two points of the cluster(C). C is defined as set of crew and M is set of maintenance tasks. $k = v = \{1, .., c\}$ are crew indices and $l = h = \{1, .., m\}$ are indices of the maintenance tasks. C_{kl} is parameter showing the distance between crew k and task $l \mid k \in C$ and $l \in M$. S_{lh} - Distance between task l and task $h \mid l, h \in M$ and d_l is duration of task l. Decision variable X_{kl} is 1 if task l belongs to cluster containing crew k otherwise 0

$$Min \quad \sum_{k=1}^{C} \sum_{l=1}^{M} X_{k,l} * C_{k,l} \; + \; \psi \; + \; w \tag{1}$$

subject to:

$$\sum_{l=1}^{M} x_{kl} = 1 \quad \forall \, k \in C \tag{2}$$

$$X_{k,l} * X_{k,h} * S_{k,l} \leq \psi \quad \forall \, k \in C \quad \forall \, l, h \in M \tag{3}$$

$$\sum_{l \in M} x_{k,l} * d_l \; - \sum_{l \in M} x_{v,l} * d_l \quad \leq \; w \quad \forall \, k \in C \quad and \quad \forall \, v \in C \tag{4}$$

3. Proposed clustering hyper-heuristic framework

In this paper, we propose a perturbative hyper-heuristic framework for solving clustering problems as illustrated in Figure 1. Like the most of perturbative hyper-heuristic frameworks, search is executed with a single candidate solution. Such hyper-heuristics, iteratively, attempt to improve a given solution throughout two consecutive phases: heuristic selection and move acceptance. A candidate solution (St) at a given time (t) is changed into a

new solution using a selected heuristic. Then, a move acceptance method is employed to decide whether to accept or reject the altered solution.

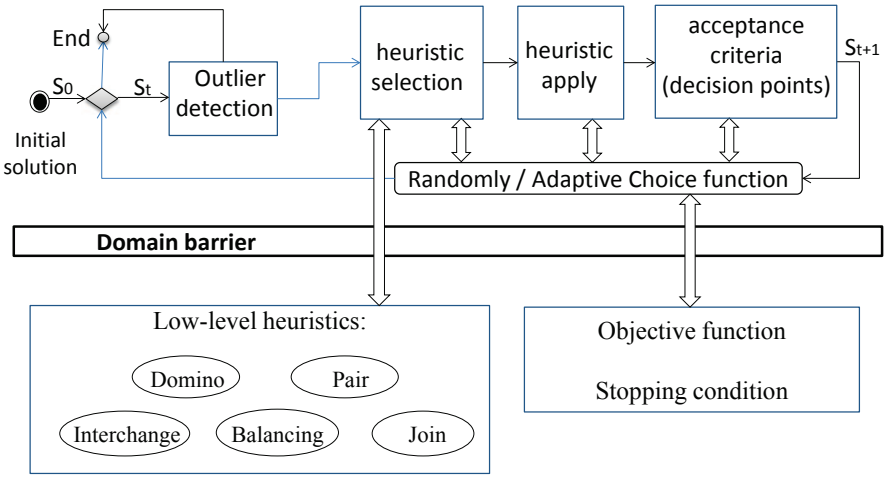


Figure 1: Proposed perturbative clustering hyper-heuristic framework

In the proposed framework, the clusters are improved by means of detecting faraway tasks from cluster center within clusters and try to assign them to a better cluster. The faraway tasks are representative of the concept of outliers in a clustering problem explained in more details in Section 4.3. The algorithm starts with an initial solution generated in a separate phase. Next, at each iteration, the algorithm tries to detect the outlier of a chosen cluster. If no outlier was found for any of the clusters of the current solution, the algorithm terminates and the best solution is returned as the final solution. But if an outlier was detected, the algorithm continues by optimizing the solution within the hyper-heuristic framework.

Five new low-level heuristics are introduced to be selected and applied in the framework. Once an outlier was found, one of the heuristics is selected to improve the solution by removing the outlier and assigning it to the best possible neighboring cluster. In this framework, the heuristic selection is being done based on an adaptive choice function which acts as a high level strategy to guide the framework to a better search space. Choice function provides the high level strategy with knowledge of the problem domain. The high level strategy selects one low-level heuristic at each decision point according to the information obtained from the feedback mechanism. The main components of the framework are described as follows.

3.1. Initial solutions

To generate initial solutions for our hyper-heuristic framework, we propose an heuristic which use an ordering strategy to assign the objects to the clusters. Table 1 represents the pseudo-code of the algorithm. It takes input parameter k which shows number of crew and partitions a set of n number of maintenance tasks into k number of crew.

```
1.   order the task list according to farthest-first strategy
2.   Initial tabuList ← empty
3.   Do
4.     If the number of clusters equals to the size of Tabu list
           empty the tabuList
     //assigning the task with the farthest distance to its closest crew:
5.     t ← taskList(n)
6.     c ← set c as the closest crew of t
7.     If (c is in tabuList)
8.           c ← a nonTabu crew
9.     assign t to crew c
10.    remove t from taskList
11.    add crew c to tabuList
12.  While(taskList is not empty)
```

Table 1: ordering heuristic for generating initial solutions

The algorithm follows a dynamic way to classify a given number of maintenance tasks through a certain number of clusters/crew. The algorithm starts with a sorted list of the tasks according to the distance of each task location to all the crew start location. The task with the farthest distance from its closest crew is placed in the first position in the sorted list meaning that algorithm gives higher priority to the farthest task to be assigned to its closest crew. In this way, algorithm prevents lately assigning of the far away tasks to the farther depots with higher distance cost. The motivation is due to balancing the workload of the crew, in terms of number of tasks assigned, there might not be any close crew left by postponing assignment of the far away tasks.

From the crew point of view, to prevent assigning the far away tasks just to a single crew, a tabu list keeps tracking on the crew assigned task at each assignment. Then, in the algorithm, if there was a task which should be allocated to the same crew recently received a task, the algorithm stops

Dataset	TotalDistance	MaxDistance	AVG_MaxDistance
E100	5546.58	230.97	115.34
E500	25568.81	166.12	109.32
E1000	51971.83	165.99	113.80
E5000	260743.92	241.39	175.65
M100	5401.86	228.03	110.16
M500	31378.27	230.48	161.86
M1000	55425.76	230.97	163.66
M5000	280743.47	233.46	166.61
R100	7526.52	236.84	152.11
R500	33290.35	233.84	161.86
R1000	64619.51	236.67	167.45
R5000	333591.41	236.20	169.75

Table 2: Results of the initial solution by ordering heuristic

assigning this task to the crew. Then it selects a non recently visited crew to assign the task to it, in turn. Hence, we take care of balancing the number of tasks assigned to each crew, gradually during construction of the solution.

It is believed that firstly assignment of the tasks with farther distance from the closest crew, generates better clustering compared to assigning the tasks in a greedy way (Assigning the tasks to their closest crew first), despite of some penalty. In fact, using Farthest First ordering, we start assigning from the most difficult tasks towards the easiest ones to the crew.Therefore, the algorithm penalizes the solution at the early steps of the algorithm, more than using a greedy approach, but at final steps (last tasks) the solution will survive from receiving high penalties from assigning remaining far away tasks to the crew.

Table 2 summarizes the clustering result obtained by the ordering heuristic on 12 datasets. The datasets represent the geographical points in the maintenance area introduced in section 4. The table shows three different measurements. First is the total distance cost which is total spatial distance of locations of the assigned tasks to the location of each crew as the cluster representative. The calculated cost here is not the traveling distance cost but the routes going to be generated in scheduling phase later on, will be a subset of total distance cost in clustering phase. Therefore we aim at minimizing this cost as one of the main cost of any maintenance planning. Second, to assess the maximum distance availability of the crew in future

breakdowns, the farthest two tasks assigned to a crew within the whole solution is calculated. This measurement is used to indicate the maximum distance where a crew can be far from the failures, if it may be responsible for future breakdowns happening within its cluster. Last, to have a better understanding of the solution state in possible future breakdowns, we calculate the average of the maximum distance traveled by each crew.

In the result section, these solutions are tested as inputs to the clustering hyper-heuristic framework with and without learning mechanism and the effect of the framework will be discussed on the initial solutions.

3.2. Handling outliers

An outlier is an observation of the data that deviates from other observations such that it arouses suspicions that it was generated by a different mechanism from the most part of data [30]. They are typically further away from the mean and are thus shifted a larger distance due to the shrinking. Outlier detection methods aims at finding of the rare data with exceptional behavior compared with other data, because they would affect the result more significantly.

In the region splitting problem described, in order to deal with the unexpected failures and breakdowns happening in the Danish railway network, we make the crew in each cluster responsible for any breakdown happening within their own cluster. In order to keep the crews available within the time limit requested by stakeholders, the maximum distance among the tasks should be minimized within each cluster. This exactly reflects the definition of the diameter of a cluster as the maximum distance between any two points of the cluster(C).

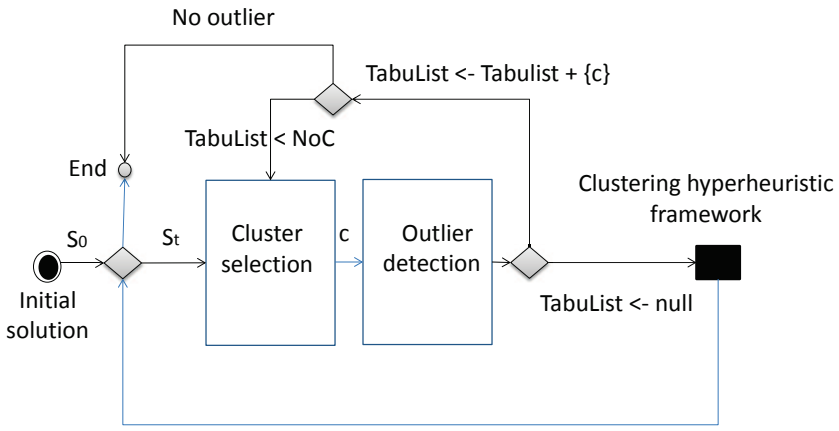


Figure 2: Module of outlier handling

One way to find and remove the outlier in this problem, is calculating the diameter of the cluster by running time of $O(n^2)$ for $n \in C$. Doing this, we find the two farthest tasks from each other in a cluster and remove the task with farther distance from cluster center, as outlier from the cluster.

To have less time complexity, we suggest using the radius of the cluster instead of diameter to detect the outlier by running time of $O(n)$ for $n \in C$. The radius of a cluster is the maximum distance between all the points and the centroid. A cluster centroid is typically the mean of the points in the cluster. Although, the radius and diameter of a cluster are not related directly, as they are in a circle, but there is a tendency for them to be proportional [31].

We may choose the radius of the cluster which is the farthest task from cluster center as outlier and continue until the radius(farthest task from the centroid) is not farther than half of the maximum allowed distance of a crew to the breakdown within its cluster diameter of the cluster). In this way, every task/point farther than radius are considered as outlier and it can be assigned to the best possible cluster. Hence, the algorithm will make sure that maximum distance from crew location to the the breakdown location could be twice as the radius of the cluster in the worst case.

To make sure the reachability of the crew within the defined time limit, we improve the clusters by removing the outliers and assigning them to the best possible choice available among the clusters.

Figure 2 shows the outlier module in the proposed framework. Once the initial solution was constructed, firstly, a cluster of the solution is selected randomly. This module attempts to find an outlier of the selected cluster by

finding a task having distance from its crew location more than half of the allowed distance between the two farthest tasks (diameter of the cluster). To avoid re-selecting the same clusters without any outlier, we have defined a tabu list. The newly chosen clusters without any outlier are added into the tabu list which has a length of clusters number. Therefore, if a cluster with outlier was found, the tabu list would be empty and the algorithm enters the second phase of improvement by the hyper-heuristic framework. Otherwise, the algorithm adds the selected cluster to the tabu list and keeps selecting a non tabu cluster until it finds either a cluster with outlier or no more non tabu cluster to choose.

3.3. Low-level heuristics

Within our clustering hyper-heuristic, we introduce five heuristics shown in Figure 3 to improve the clusters at each iteration of the algorithm. The circles represent the clusters and the points indicate tasks/objects within that particular cluster. Red points are outliers while the black points could be either an outlier or a normal object/task. These low-level heuristics are used and their performances are discussed in the proposed clustering hyper-heuristic framework both with randomness selection and chosen by the choice function:

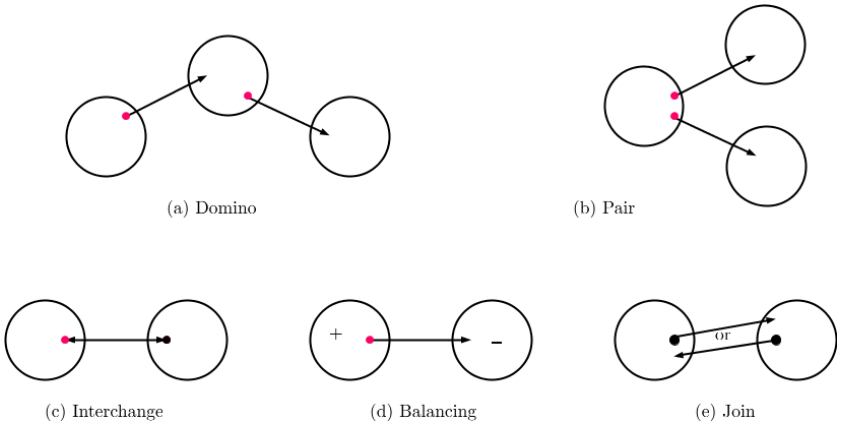


Figure 3: Proposed Low-level heuristics

Domino: This heuristic works like a domino game but only in two steps such that removing the outlier from current cluster and assign it to the next cluster leads to the same trend on the second cluster. Domino heuristic removes the outlier of the current seed cluster whether selected randomly or by learning mechanism and gives it to the cluster with the closest center

distance to the outlier. Consequently, the cluster which has received the outliers does the same and removes its farthest object(outlier) to another cluster with the closest center distance to the outlier. The second assignment could return to the first cluster which leads to have a balanced exchange.

Pair: This heuristic removes two outliers sequentially from the current seed cluster and assign them to the best possible cluster in terms of distance of the outlier to the candidate clusters centers. The destination cluster for the two outliers could be same or different. This heuristic changes the balance of the clusters.

Interchange: This heuristic tries to give the outlier to the closest cluster and get an object back from the same cluster, in turn. The object from the second cluster could be an outlier(in the best case) or any other object with closer distance to the center of the first cluster than its own. The order of finding the right choice from the second cluster is sorting order of N/C object where N is number of whole tasks and C is number of the clusters. The benefit of using this heuristic is that it keeps an eye on improvement and at the same time balances the clusters. However, due to running time, we could not limit our methods/low-level to use only this type of strategies.

Join: This low-level looks for adjacent objects/tasks which belong to the different clusters and tries to join these two tasks together by assigning one of them to the cluster of the other task. The decision is made based on average closeness of the two objects from the centers of the clusters.

Balancing: This heuristic tries to remove the objects from clusters with extra objects and assign them to the clusters with less objects to have a balanced clustering at the end.

All of the proposed low-level heuristics except balancing have been defined as improvement methods. It means that when they are chosen to apply on the solution, the solution either is improved or remained the same by the current chosen low-level. While balancing low-level does not care about its effect on the objective function and it aims only at balancing the current selected cluster in the current solution.

3.4. Choice function

Different low-level heuristics have different impacts depending on their characteristics and the time they are applied to the solution space. Hence, a mechanism to recognize the nature of the solution space and consequently apply the appropriate heuristic at each step can persuade the solution to a better region in the solution space. The choice function is a smart selection method which scores heuristics based on their performance statistics.

At each decision point in the framework, the choice function is calculated based on a combination of three different measures considering the improvement the low-level heuristic makes to the clustering solution and the time taken to achieve that improvement. The choice function uses the information about the individual impact of each low-level heuristic(f1), joint impact of pairs of heuristics(f2), and the amount of time elapsed since the low-level heuristic was last called(f3). The first two parts of the choice function maintain the exploitation of the search space by collating the information of both individual and pair performance of the heuristics. While the third part of the formula takes care of exploration by selecting the heuristic that has not been called recently. At each decision point, the choice function is evaluated for all heuristics, and the heuristic with the highest score is selected for the next iteration:

$$f(h_j) = \alpha f_1(h_j) + \beta f_2(h_k, h_j) + \gamma f_3(h_j) \tag{5}$$

in more details:

$$f(h_j) = \sum_n \alpha^{n-1} \frac{I_n(h_j)}{T_n(h_j)} + \sum_n \beta^{n-1} \frac{I_n(h_k, h_j)}{T_n(h_k, h_j)} + \gamma\tau(h_j) \tag{6}$$

Where $I_n(h_j)$ (respectively $T_n(h_j)$) is the change in the evaluation function (respectively the amount of CPU time taken) the n^{th} last time heuristic h_j was called. $I_n(h_k, h_j$(respectively $T_n(h_k, h_j)$ is the change in the evaluation function (respectively the amount of CPU time taken) then n^{th} last time heuristic h_j was called immediately after heuristic h_k. $\tau(h_j)$ is the number of seconds of CPU time which have elapsed since heuristic h_j was last called.

To enhance the generality and the robustness of our clustering hyper-heuristic framework, it is desirable to apply a self-adoptive hyper-heuristic which is able to adjust itself to the condition of its operating area(e.g. heuristic space, solution space). One way to achieve self-addictiveness in the choice function is to maintain an adaptive ranking of the heuristics based upon which the hyper-heuristic makes an appropriate heuristic selection. In this way, rather than applying a choice function with constant parameters (α, β and γ) during the search, we apply the adaptive choice function based on that of Soubeiga [32] that adaptively adjust the choice function parameters at each decision point. Consequently, the framework no longer needs parameter specification from the user side.

3.5. Pseudo-code of the proposed framework

The clustering hyper-heuristic framework that we present in this paper is composed of three phases: generating initial solution, detecting the outlier and improvement of the solution through the hyper-heuristic framework. In each run of the algorithm one initial solution is generated and then the solution is improved by collaboration of outlier detection and improvement hyper-heuristic phases.

Table 3 presents the pseudo-code of the proposed hyper-heuristic approach for clustering problem. The search space of the high level heuristic consists of all of the possible permutations of the low-level heuristics defined in Section 3.2. The algorithm starts with generating a solution using ordering heuristic and the initialization of the low-level heuristics (Line 1 and 2). Once the solution was constructed, the algorithm enters the main loop to find an outlier of one of the clusters and improve the solution iteratively until the stopping condition met. Outlier detection(Line 6) has been explained in details in section 3.2.

```
1.  generate initial solution
2.  initialize clustering heuristic list hl = h1 h2 h3 h4 h5
3.  N: No of tasks in dataset, C: No of clusters, iteration = 0
4.  Do
5.    NoImprovment = 0
6.    Detecting the outlier
7.    If (any outlier was found)
      {
8.      Compute choice function for each heuristic
9.      Select heuristic hj for which f is max
10.     If (NoImprovment is ≤ N)
11.       if (NoImprovment = N)
12.         select H2 for which f3 is max and H2 ≠ hj
13.       else
14.         Select hi for which f-f3 is max
15.       Identify biggest contributor G
16.       if (G = f3 or NoImprovment = N)
17.         Select hi for which f-f3 is max and Apply in steepest descent
18.         if (there is any improvement and hi ≠ hj) Punish G
19.       if (G = f1 or f2)
20.         Apply hj in steepest decent , Punish or Reward α or β
21.       if (G ≠ f1 or f2 or f3)
22.         Apply hj in steepest decent
23.     Calculate Absolute improvement
24.     update NoImprovement (zero or NoImprovement + 1 )
25.     If (NoImprovment > N)
26.       Reward f3 , Apply H2 in steepest decent, NoImprovement = 0
27.       if (there is no improvement) undo steepest decent and apply H2 once
      }
28    iteration = iteration + 1
29  While (stopping condition met)
```

Table 3: Pseudo-code of the proposed perturbative clustering hyper-heuristic framework

After initial solution construction, if an outlier was found, the algorithm starts improving the solution through the proposed clustering hyper-heuristic framework. As explained earlier, to enhance the robustness of the presented framework in this paper, we employ the adaptive choice function which is able to adjust itself to the condition of search space it is operating in. At the beginning of the search, variable $NoImprovement$ is defined to

keep track of number of consecutive iterations which makes no changes in the objective function. Then the choice function value is computed for each heuristic to select the one with highest f value (Line 8 and 9). In order to determine whether we need to exploit or explore the solution space at each iteration, the biggest contributor of the highest f is distinguished. This prescribes the way the chosen heuristic is applied.

In general, when the algorithm recognizes that we are in an exploitation phase ($G = f1$ or $G = f2$), the chosen heuristic is applied in a steepest decent fashion (line 19 and 20). If the solution needs the exploration ($G = f3$), the chosen heuristic is applied once. But as suggested in the adaptive choice function, it is better to be applied in a steepest decent way if it led to improvement otherwise the solution is returned to the previous solution and the heuristic is applied once (line 16 to 18). $H2$ is the heuristic used to set up an appropriate level of exploration when both of $f1$ and $f3$ are maximum[32].

The parameters of α , β and γ will be rewarded or punished if the resulting solution is better or worse than the previous solution, respectively. This adaptivity in the framework allows interplay between the parameters of the choice function regularly by modifying the weighting assigned to each parameter according to the performance of the each low-level heuristic.

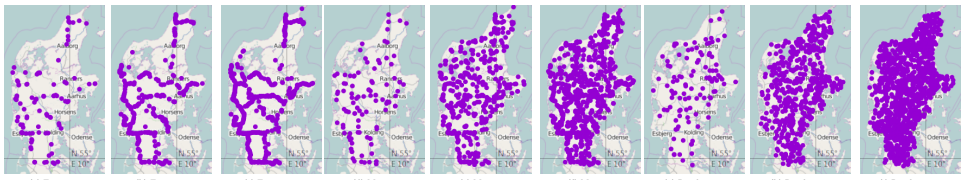
The algorithm stops under three different criterion. First, when no outlier was found either in the generated initial solution or in the improved version of solution when all of the outliers are assigned to a better cluster choice. Therefore if we make an initial solution without outlier, the solution is returned unchanged through our framework.Second, when the algorithm has detected an outlier but the hyper-heuristic could not improve the solution after a certain number of iteration. In this condition, we consider 0.1 of number of the tasks as stopping criteria. Lastly, if the algorithm didn't fall under the previous conditions, the framework stops when a given threshold(number of tasks in each dataset) is achieved by consecutive iterations.

4. Datasets

In this section we define test cases used in the experimental results section. The geographical points are all located in the Danish peninsular of Jutland. To standardize our test cases, we follow the file format from classical benchmark test sets for Vehicle Routing Problem with Time Windows (VRPTW) introduced by Solomon in 1998 (http://w.cba.neu.edu/ msolomon/problems.htm). The tasks should be assigned to a number of crew members. The coordinates representing the geographical location of the tasks generated by

utilizing the Google Map API. This is done based on three different task locations:

1. Tasks are all located on the rail tracks of maintenance area.
2. Tasks are located randomly on or off-track.
3. Tasks are scattered around the whole area randomly.



(a) Exact100 (b) Exact500 (c) Exact1000 (d) Mix100 (e) Mix500 (f) Mix1000 (g) Random100 (h) Random500 (i) Random1000

Figure 4: Geographical Visualization of the Dataset

Figure 4 represents a geographical visualization of the datasets. Each set of test cases has four different number of tasks which are: 100, 500, 1000 and 5000. These should be serviced by a team of 8 crew members. These numbers are chosen respectively according to the number of maintenance tasks which needs to be done on a daily, weekly, monthly and annually bases. The datasets and the documentation about how the datasets has been created are accessible at https://github.com/ShahrzadMP/Dataset.

5. Results and discussion

This section reports the result of the experiments carried out in this study to present the proposed clustering hyper-heuristic framework. Initially, the results of the clustering hyper-heuristic is compared with the results of formulating the problem as Assignment Problem(AP) and with the Simplified assignment algorithm which is a heuristic approach. Then the proposed framework with and without learning mechanism is compared to the results of the initial solution by ordering heuristic and the ability of the low-level heuristics in design level are discussed. Third, the performance of the heuristics are analysed through the frequency of their calls on the three biggest datasets and then the heuristics are ranked according to their performance result. Forth, the trend of improvement by the Adaptive clustering hyper-heuristic framework is shown on the $E5000$ dataset. Finally, we compare the cohesion of the clusters result obtained by the ordering heuristic, RCHH and ACHH are calculated and compared together to see the effect of partitioning on scheduling phase.

5.1. Comparison with previous approaches

The purpose of this set of experiments is to making a comparison between the proposed clustering approach with two other exact and heuristic assignments algorithms from the literature. As mentioned earlier, in cluster-first, route-second approaches, the clustering part of the approaches for MDVRP is usually solved by mainly assignment algorithms. Assignment Problem(AP) as special case of Transportation Problem is the only exact approach introduced to give an assignment of the customers to depots[33].

Our heuristic candidate to compare our obtained results is the Simplified assignment[16]. This algorithm is one of the common algorithms as subclass of urgency algorithms[34]. Urgency algorithms are recommendable for big real life problems due to their medium execution time and the good results compared to the other heuristics[17]. In the simplified algorithm, the comparison is between the cost of assigning a customer to its closest depot with the cost of assigning it to its second closest depot.

Since AP is only concerned with finding the minimum cost of transportation from depots to the customers, it is not fair to compare AP with our approach without taking the balancing of the depots/crew into account. Hence, to have a fair comparison between AP and our proposed approach, we have added the balancing as a second objective function to the AP. In this way each depot/crew needs to be given approximately the same amount of work. The transportation problem is therefore formulated as follows:

K is set of crew(depot), M is set of maintenance tasks/customers. C_{kl} is distance between crew/depot k and task(customer) l such that $k \in K$ and $l \in M$. d_l is duration of task(demand) l.

x_{kl} is decision variable which is 1 if task l belongs to the cluster containing crew k, and otherwise is 0.
w is positive variable representing an upper bound for maximal workload difference between crew pairs in terms of task duration.

$$Min \quad \lambda * \sum_{k \in K} \sum_{l \in M} x_{k,l} * C_{k,l} + \quad (1-\lambda) * w \tag{7}$$

subject to:

$$\sum_{k \in K} x_{kl} = 1 \quad \forall\, l \in M \tag{8}$$

$$\sum_{l \in M} x_{kl} * d_l - \sum_{l \in M} x_{vl} * d_l \quad \leq w \quad \forall\, k \in K \quad and \quad \forall\, v \in K \tag{9}$$

The objective function (7) becomes multi criteria and aims to find the optimal trade-off between assigning tasks to the crew based on their spatial proximity while taking the balancing crew workload into account. The second term in the objective function, w, is the upper bound for work load balancing mismatches across different clusters as described by constraint (9). The weights assigned to the two terms of the objective function are given as λ and $1 - \lambda$. For our numerical results presented in this paper we have chosen (0.3) and (0.7) on the first and second term respectively. By applying the weighted sum approach, this gives a reasonable trade-off between work balance and the total distance covered. Constraint (8) imposes that each task should be assigned to only one crew. To compare with the Assignment Problem, the model was coded in GAMS and run on 2.10GHz, 8.00GB RAM, 64-bit operating system, Intel (R) Core (TM) i7-4600U CPU.

Table 4 shows the results obtained by AP (by taking the crew/depot balancing into account), Simplified assignment and ACHH on three different measurement; The total distance of task locations to the crew location in each cluster(TD), the maximum distance of a crew from the unexpected failures(MDD) and the average of the maximum distance(AVG_MDD). Each set of results obtained by AP and S_A is followed by a column showing the percentage of the ACHH improvement on corresponded measurement in comparison with AP and S_A results, respectively.

Comparing ACHH result with AP as an exact approach, we can see that the quality of solutions obtained by ACHH is approximately within 10% improvement gap with AP results in TD and AVG_MDD. Looking at MDD, ACHH could obtain better maximum distance on the railway track dataset($E100, E500, E1000$), while AP gained better results at MDD on the two other sets of instances ($M100, M500, M1000$ and $R100, R500, R1000$). It is worth to mention that MDD only highlights the worst cluster among the others and AVG_MDD is a better representative than MDD by calculating the average of maximum distance for all of the clusters. Regarding the size of data solvable by AP and ACHH, GAMS could not solve AP for the biggest data instances with 5000 number of tasks.

Table 4: Comparison of the result found by Assignment Problem(AP), Simplified Assignment(S_A) algorithm and Adaptive Clustering Hyper-Heuristic(ACHH) on three different measurements of total distance(TD), maximum distance availability(MDD), and the average of the maximum distance(AVG_MDD)

Dataset	TD				
	AP	%	S_A	%	ACHH
E100	4596.11	-3.94	5233.94	8.73	4777.15
E500	22554.32	-4.06	25460.18	7.81	23470.77
E1000	46168.23	-0.28	51901.78	10.80	46297.18
E5000	X	X	260694.49	8.24	239216.47
M100	4315.46	-6.58	5154.75	10.78	4599.21
M500	26781.42	-1.10	31302.68	13.50	27075.32
M1000	49203.80	3.65	55317.69	14.30	47407.02
M5000	X	X	280666.80	12.85	244597.76
R100	6564.27	1.42	7413.68	12.71	6471.16
R500	30369.95	5.17	33214.83	13.30	28798.41
R1000	58270.62	8.27	64545.90	17.19	53450.73
R5000	X	X	333471.23	10.61	298091.19
	MDD				
	AP	%	S_A	%	ACHH
E100	156.53	12.04	255.07	46.02	137.69
E500	156.04	0.90	189.40	18.36	154.63
E1000	166.43	26.23	190.00	35.38	122.77
E5000	X	X	265.03	29.82	186.01
M100	149.08	-35.83	248.63	18.55	202.50
M500	202.77	-6.19	254.40	15.36	215.33
M1000	167.18	-20.65	258.02	21.82	201.71
M5000	X	X	257.41	14.15	220.98
R100	192.37	-20.47	245.83	5.73	231.74
R500	201.61	-28.27	259.71	0.42	258.61
R1000	209.73	-22.87	264.88	2.71	257.69
R5000	X	X	265.10	-6.36	281.97
	AVG_MDD				
	AP	%	S_A	%	ACHH
E100	94.72	-2.85	143.78	32.24	97.42
E500	94.66	-9.87	138.19	24.74	104.00
E1000	102.37	9.66	142.47	35.09	92.48
E5000	X	X	208.30	40.86	123.18
M100	95.77	-3.98	143.31	30.51	99.58
M500	126.57	-9.28	193.42	28.49	138.31
M1000	118.47	0.81	197.56	40.52	117.51
M5000	X	X	197.96	22.22	153.97
R100	136.81	-5.96	166.05	12.70	144.96
R500	139.16	-11.06	184.87	16.40	154.55
R1000	139.93	-8.18	197.26	23.26	151.37
R5000	X	X	195.81	21.74	153.24

ACHH in comparison with Simplified Assignment, gave an improvement of 11%, 17% and 27% in terms of TD, MDD and Avg_MDD, respectively. We believe that the notable improvement on MDD and Avg_MDD compared to TD is attributed to the nature of proposed hyper-heuristic framework which mainly improves MDD of the clusters through reshuffling the outliers.

5.2. Clustering hyper-heuristic framework

The second set of experiments aimed at making a direct comparison between the random clustering hyper-heuristic (RCHH), and the adaptive choice-function based clustering hyper-heuristic (ACHH). Results (best over 5 runs) are given in Table 5 for 12 instances. Both RCHH and ACHH starts with a solution produced earlier by an ordering heuristic.

The table shows the total distance of task locations to the crew location in each cluster, the maximum distance of a crew from the unexpected failures and the average of the maximum distance traveled by each crew. Each of these measurements is followed by a column showing the percentage of the improvement on corresponded measurement compared to initial solution results shown earlier in Table 2. The last row of the table represents the average percentage of the improvement achieved by RCHH and ACHH on overall dataset.

From Table 5 we can see that both RCHH and ACHH improved the results of the heuristic algorithm. We believe this happens by rationale behind the proposed low-level heuristics; Domino, Pair and Join which minimize the maximum distance of a cluster, accordingly, minimize the overall distance of a clustering result through reassigning the outliers to a better cluster. These heuristics helps intensification of the search space by focusing only on minimizing the total distance in order to give a better solution with less cost. Interchange which tends to both minimize the total distance and keep the balancing of the clusters, tries to intensify the search space like the others, despite that it does not affect the balancing state of the solution. Finally, Balancing heuristic just takes the balancing of the clusters into account. The effect of this heuristic is basically the diversification of the search space in order to avoid getting trapped in a local optimum. However, there will be also possibility of exploitation the search space if it leads to a solution with less total cost compared to previous solution. The obtained results proves that although the effect of these methods are very dependent on when and how long they are applied to a solution in the framework, they still have been designed in an appropriate way to be able to explore different areas of the search space, effectively.

	TotalDistance				MaxDistance				Avg_MaxDistance			
	RCHH	%	ACHH	%	RCHH	%	ACHH	%	RCHH	%	ACHH	%
E100	5337.73	3.77	4777.15	13.87	166.12	28.08	137.69	40.39	109.15	5.37	97.42	15.54
E500	25143.12	1.66	23470.77	8.21	176.43	-6.21	154.63	6.92	113.07	-3.43	104.00	4.87
E1000	51156.28	1.57	46297.18	10.92	174.74	-5.27	122.77	26.04	121.50	-6.77	92.48	18.73
E5000	258495.71	0.86	239216.47	8.26	192.68	20.18	186.01	22.94	137.41	21.77	123.18	29.87
M100	4980.06	7.81	4599.21	14.86	202.50	11.20	202.50	11.20	110.85	-0.63	99.58	9.60
M500	30131.16	3.97	27075.32	13.71	220.95	4.13	215.33	6.57	157.03	2.98	138.31	14.55
M1000	52698.16	4.92	47407.02	14.47	208.75	9.62	201.71	12.67	145.62	11.02	117.51	28.20
M5000	270948.83	3.49	244597.76	12.87	217.10	7.01	220.98	5.35	157.72	5.34	153.97	7.59
R100	7184.99	4.54	6471.16	14.02	272.11	-14.89	231.74	2.15	159.43	-4.81	144.96	4.70
R500	32587.91	2.11	28798.41	13.49	277.53	-18.68	258.61	-10.59	181.21	-11.95	154.55	4.52
R1000	61558.06	4.74	53450.73	17.28	225.64	4.66	257.69	-8.88	159.94	4.48	151.37	9.60
R5000	324239.05	2.80	298091.19	10.64	236.20	0.00	281.97	-19.38	162.84	4.07	153.24	9.73
Avg		**3.52**		**12.72**		**3.32**		**7.95**		**2.29**		**13.12**

Table 5: Result of Adaptive and Random Clustering hyper-heuristic on initial solution

As Table 5 shows, ACHH return significantly better results than RCHH. Interestingly, the results correspond to the best value of total distance and average of maximum distance found by adaptive hyper-heuristic for all of the dataset comparing to random one. ACHH yielded roughly 13% improvement in both total distance and average of maximum distance and 8% in maximum distance at the final clustering result. While RCHH improved the heuristic results approximately only 3% on all of the three measurements.

Since we use the same low-level heuristics in both frameworks, the significant improvement of ACHH than RCHH is owing to the self-adaptiveness ability throughout the hyper-heuristic search. This characteristic takes care of the proportion of exploitation/exploration appropriately by adjusting parameters α, β and γ in every iteration. While, in RCHH, choosing the low-level heuristics randomly may lead the solution to the area of the search space at are difficult to quickly move to another area. For instance, applying those proposed low-level heuristics which only pay attention to minimizing the distance without taking care of balancing of clusters such as Domino, Pair or even Join, might lead the space to an area with very high quality in terms of overall total distance and the maximum distance but very low quality in balancing. In this situation, moving the solution space back to a space resulting in a balanced solution might cause paying too much penalty in the objective function.

We note that the gap in terms of TD, MDD and Avg_MDD between ACHH and RCHH is greatest with R1000(+12.5%), E1000(+31%), E1000(+25%)

and smallest with E500(+6.5%), M100 (0%), M5000(+2.5%), respectively. Contrary to expectations, it seems that though ACHH outperforms RCHH in dataset with size of 1000, but no clear relation appears as the size and type of the instances to solve varies. This might be associated with the nature of dataset which are only different from each other in terms of size and location of the tasks in the area(on the rail tracks, out of the tracks and mix of them). Due to nature of distance based objective function, consequently, the nature of presented low-level heuristics, the result should not be dependent on the dataset, since the datasets are different only in terms of geographically located in different places. Because the distance factor is consistent despite of the location of objects.

Regarding the size, the result shows no general rule associated with growth of the size of dataset. For instance, the results obtained on E100, M100 and R100 shows values of 13.87%, 14.86% and 14.2% on Total distance which is over the average of the total distance(12.72%) on whole dataset. Similarly, on maximum distance E100 achieved 40% and M100 obtained 11.20%. We believe that this may be caused by ability of ACHH in learning at early stages which makes the framework more robust and stable even in small instances. In other words, adjustment of the parameters α, β and γ in adaptive manner may speed up the convergence of a hyper-heuristic without knowledge at first iterations to a learning framework as iteration increases.

To analyze this, in the next set of experiments, we deal with the low-level behaviour of the ACHH and the frequency of the heuristic calls when the iteration increases in the framework.

5.3. Low-level heuristics performance

In Table 6 we give the ratio of call, by ACHH, of each clustering low-level heuristic during the first 100 heuristic calls and during the last 100 heuristic calls to the best solution of biggest dataset(e.g E5000, M5000 and R5000). From the ratio of calls during the 1st 100 calls it is clear that in the early stage of the search calls are not spread as moderately as the last 100 calls of the search. It is interesting that Domino has the most call(83,60 and 50) and Balancing(2,3 and 1) has the least call during the 1th 100 calls in all three dataset. This indicates that the framework still can recognize the improvement heuristic and the destroying one in terms of minimizing the distance even in the early stages. However, applying Domino heuristic which only affect on improvement of the total distance is still indicator of a greedy behavior of the framework at this stage. Therefore, it is interesting to analyse how the learning progresses as the iteration increases.

	E5000	M5000	R5000
Balancing	**2**/19	**3**/15	**1**/21
Domino	**83**/23	**60**/16	**50**/20
Join	9/23	18/37	1/20
Interchange	6/16	5/16	1/19
Pair	0/19	14/16	47/20

Table 6: Heuristic calls during 1st 100 calls/during last 100 calls to the solution by ACHH

E5000		M5000		R5000	
787/10000	**5**	1207/10000	**5**	819/5650	**5**
1253/10000	2	1539/10000	4	940/5650	4
5883/10000	**1**	3987/10000	**1**	1927/5650	**1**
990/10000	4	1682/10000	2	1012/5650	2
1087/10000	3	1585/10000	3	952/5650	3

Table 7: Overall proportion of call and overall rank of designed heuristics by ACHH

From the last 100 calls it is noticeable how the propagation of calls over the heuristics converge to the more fairly way when the size of dataset grows. It is interesting to note that all the low-level heuristics are called at this late stage which express how good the low-level heuristics are designed and combined in such a way that all of the them are almost equally involved until the final steps of the algorithm. Since the three problem instances share almost the same distribution trend of the low-level heuristics in the last 100 calls, it seems that ACHH shows the same behaviors for different problem instance. As discussed earlier in this section, we still believe that it is because of the dataset are different only in the form of distribution of the tasks and the size of dataset.

In Table 7, we rank the clustering heuristics according to their overall ratio of call so that the top (bottom) heuristic is the one that has been called most (least) often. In dataset E5000 and M5000, the stopping condition has been the third stopping criteria when algorithm overrun twice of the size of dataset (here 2×5000) explained in section 3.5. But R5000 stops in either first or second stopping condition because the iteration number(5650) is less than the iteration threshold (10000) in the third stopping condition.

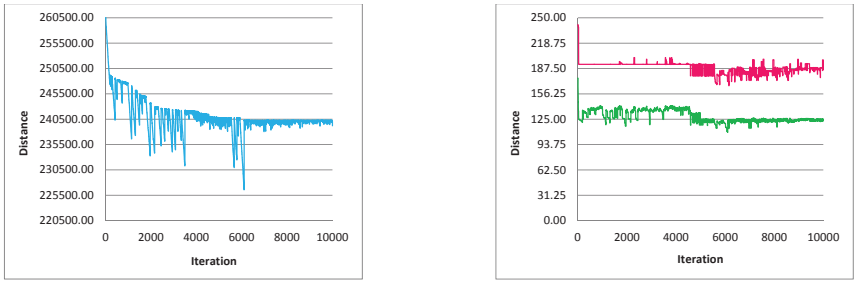
From the overall ratio of calls we see that overall (across the 3 instances), heuristics Join and Interchange appear among the top two heuristics whereas heuristic Balancing is at the bottom. This can be regarded as a feature common to the Join and Interchange which explore the solution space in different way comparing the other designed low-level heuristics. Join is the only low-level that tries to minimize the total distance not by dealing with the outliers but by joining the close tasks of the different clusters. There could be many tasks in close neighborhood but belonging to different clusters which can be joined to the same clusters and improve the total driving distance differently. Specifically, when the algorithm cannot improve the solution by dealing with the outliers, whether the best assignment is the

current cluster or the solution space gets stuck in a local optimum.

Interchange is designed in a way that not only improves the solution without being limited to dealing with the outliers but also takes care of balancing factor in the solution space. Regarding the rank of Balancing heuristic, it is not a surprise, since it doesn't care about minimizing the total distance as the objective function at all. However, the number of calls for Balancing shows that how the parameter γ has been appropriately tuned to explore the search space by calling the forgotten Balancing heuristic during the search.

5.4. ACHH improvement trend

Figure 5 shows the trend of improvement of the best solution on dataset $E5000$ over one run. The y-axis in (a) is the total cost of driving distance. Similarly, it is the maximum distance of a crew to a cluster(Red plot) and average of the maximum distance obtained by all of the crew over the iterations in (b). Because the choice function has shown almost the same trend in all big instances, only trend of one dataset is being investigated.



(a) distance distance (b) MaxDistance and AvgMaxDistance

Figure 5: Trend of improvement of $E5000$ solution over the iteration

It is evident that ACHH showed overall downward trend in minimizing the distance throughout the iterations. In early iterations, it seems that ACHH improves the initial solution fast means that the ACHH does not much time to start improving the solution. However the best solution fluctuated between the 1000 to 4000 iterations. One possible explanation might be due to postponing the call of Balancing heuristic since whenever it calls, it causes a bad penalty in the total distance.We think this trend could be improved by somehow considering the balancing of the framework as a value instead of calculating only the penalty of increase at the total distance. In

this way, Balancing could be called more often,consequently, led to less fluctuation comparing to the current trend. However, it is notable how the algorithm stabilizes approximately after half of the iteration passes. Similarly, the average of maximum distance (green plot) shows the same trend with a significantly drop in early iterations, following by fluctuation and finally remaining steadily by a marginal change over the average distance.

In contrast to total distance and average of maximum distance, max distance plot (red) fluctuates more in the next half of the iteration than early stage of the algorithm. This indicates how the designed low-level heuristics can fairly correlate with each other to improve all of the embedded factors (minimizing total distance, minimizing the maximum distance and the balancing of the clusters) in them, after the algorithm reach to a appropriate level of adaption through the parameters α, β and γ.

5.5. Compactness validation

As mentioned earlier, the clustering framework presented in this paper is being used as the region splitter for partitioning of the danish railway system. This phase is a prior phase to the preventive maintenance planning in the signaling system based on ERTMS technology. In this way, the planner makes sure that no far away task will be assigned to any crew in scheduling phase. Moreover, it is interesting to see how the railway network looks like from a scheduling problem point of view. Applying this partitioning, the maintenance planning in Denmark resembles the classical problem of vehicle routing problem(VRP) within in each cluster, despite having just one crew/team responsible for scheduling of the tasks. Mainly in any scheduling problem, the objective is to minimize the total cost (i.e., a weighted function of the number of routes and their length) to serve all the tasks. Therefore, the closeness of the tasks in each cluster can affect the length of routes and then the total cost in the scheduling phase.

To calculate the cohesion of the clusters, in addition to discussed results by the other problem-specific measurements, we calculate the validity factor of the compactness which is a well known measurement in the literature.

Compactness factor is measure of cohesion of the objects in every single cluster. it indicates how well a clustering algorithm partitions the objects sets in a distinct clusters in terms of object homogeneity by the mean normalized variance. In other words, this index is formulated to decide whether a given subset is internally dense or not. The higher this value is, the lower

average cohesion of the cluster is:

$$\mathcal{C} = \sum_{k=1}^{K} \sum_{i=1}^{N} P_{k,i} \left\| X_i - C_k \right\|^2 \tag{10}$$

Where $\mathcal{C}$ is the compactness value for the clusters desired to be minimized. K is the number of the clusters. N is the number of the nodes. P is the partition matrix and $P_{i,k}$ specifies if the X_i is in partition k. C_k is the center of partition k.

Figure 6 presents the comparative results of compactness measure of the initial clustering result, random hyper-heuristic framework and the adaptive framework on $E5000$. The compactness of the clustering results obtained by RCHH and ACHH are shown as ratio of compactness measurement obtained by them to compactness of the initial clustering result. As the lower compactness measurement shows the denser clusters, it is so evident how ACHH generates much more compact clusters compare to RCHH and the initial solutions. It is notable that ACHH could improve approximately 32% on compactness of the initial solution, while RCHH improved 11% of the measurement,respectively, on average in all of the dataset.

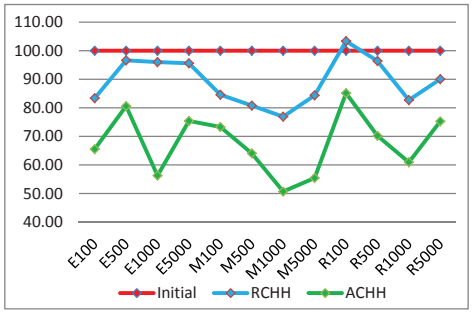


Figure 6: Compactness clustering measurement

The single highlight result is the performance of the RCHH on $R100$ dataset , when it cannot improve the compactness of the initial result by giving roughly 5% worse in compactness factor. However this was not an unanticipated outcome, when the RCHH could not gain any improvement on average(-4.81%) and maximum distance(-14.89%)the initial clustering solution shown in Table 5, earlier.

6. Conclusions

In this study, we have proposed a perturbative clustering hyper-heuristic framework. The novelty of the framework lies in the design level which improves a clustering solution through handling the outliers within a new clustering hyper-heuristic framework. Five low-level heuristics have been introduced and employed as local searchers. Domino, Pair, Join and Interchange are designed to improve the total distance. The Balancing heuristic aims for result in a more balanced clustering solution by reshuffling the outliers between clusters. The first group of heuristics intensively search the search space whilst the latter heuristic explores the search space by only attempting to give a balanced result. To have a robust hyper-heuristic framework, an adaptive choice function has been embedded as a learning mechanism to select and apply appropriate heuristics at each decision point.

The behavior of the introduced low-level heuristics using randomness and the adaptive choice function has been investigated. The designed heuristics are able to improve a clustering result even by calling them randomly which relies on their capability to exploit and explore the solution space effectively. In addition, employing the adaptive tail-made choice function, yields a more effective combination of the heuristics by selecting the appropriate probabilities for calling heuristics. The adaptive framework(ACHH) led to an improvement of approximately 10% in both total driving distance and average of maximum distance compared to the random framework (RCHH).

The experiments showed that the presented framework could be employed as a region splitter in railway systems. We suggested this partitioning as a prephrase to preventive maintenance planning. By making each crew of the cluster responsible for future breakdowns in their own cluster, the planners are able to estimate the maximum time availability of the crew when a failure happens. In the proposed framework, we try to achieve lower distance availability during failures by reassigning the outliers iteratively to a better cluster choice. Moreover, this partitioning results in rescaling the size of the problem in the scheduling phase, having the possibility to undertake parallel scheduling, and accordingly use of sophisticated and higher quality approaches like metaheuristics/hyper-heuristics to plan the maintenance tasks.

7. References

[1] P. Barger, W. Schön, M. Bouali, A study of railway ERTMS safety with Colored Petri Nets, in: G. S. . M. e. Bris (Ed.), The European

Safety and Reliability Conference (ESREL'09), Vol. 2, Taylor & Francis Group, 2009, pp. 1303–1309.

[2] A. P. Patra, P. Dersin, U. Kumar, Cost effective maintenance policy: A case study, International Journal of Performability Engineering, Int. J. Perform. Eng 6 (6) (2010) 595–603.

[3] Banedanmark., Trafikministeriet., The signalling programme : a total renewal of the Danish signalling infrastructure, Banedanmark, 2009.

[4] V. Estivill-Castro, J. Yang, Fast and robust general purpose clustering algorithms, Vol. 1886 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2000. doi:10.1007/3-540-44533-1_24.

[5] A. K. Jain, M. N. Murty, P. J. Flynn, Data clustering: A review, Acm Computing Surveys, Acm Comput Surv 31 (3) (1999) 264–323. doi:10.1145/331499.331504.

[6] J. Jacques, C. Preda, Functional data clustering: a survey, Advances in Data Analysis and Classification 8 (3) (2014) 231–255. doi:10.1007/s11634-013-0158-y.
URL http://dx.doi.org/10.1007/s11634-013-0158-y

[7] E. Kolatch, Clustering algorithms for spatial databases: A survey.

[8] L. Rokach, O. Maimon, Clustering methods, in: O. Maimon, L. Rokach (Eds.), Data Mining and Knowledge Discovery Handbook, Springer US, 2005, pp. 321–352. doi:10.1007/0-387-25465-X_15.
URL http://dx.doi.org/10.1007/0-387-25465-X_15

[9] K. S. Mwitondi, Introduction to clustering large and high-dimensional data., Journal of Applied Statistics 38 (2) (2011) 435.

[10] C. Blum, A. Roli, Metaheuristics in combinatorial optimization: Overview and conceptual comparison, Acm Computing Surveys, Acm Comput Surv 35 (3) (2003) 268–308. doi:10.1145/937503.937505.

[11] J. Han, M. Kamber, J. Pei, Data mining : concepts and techniques, Morgan Kaufmann, 2011.

[12] T. Velmurugan, T. Santhanam, A survey of partition based clustering algorithms in data mining: An experimental approach, Information Technology Journal, Vol 10, Iss 3, Pp 478-484 (2011).

[13] S. Borah, M. K. Ghose, Performance analysis of aim-k-means and k-means in quality cluster generation.

[14] W. R. Fox, Finding groups in data (book)., Journal of the Royal Statistical Society: Series C (Applied Statistics) 40 (3) (1991) 486.

[15] J. Lenstra, A. Kan, Complexity of vehicle-routing and scheduling problems, Networks, Networks, Networks, Networks an International Journal 11 (2) (1981) 221–227.

[16] I. Giosa, I. Tansini, I. Viera, New assignment algorithms for the multi-depot vehicle routing problem, Journal of the Operational Research Society 52 (5) (2001) 977–984. doi:10.1057/palgrave.jors.2601426.

[17] L. Tansini, M. E. Urquhart, O. Viera, Comparing assignment algorithms for the multi-depot vrp, Reportes Técnicos 01-08.

[18] J. Schulze, T. Fahle, A parallel algorithm for the vehicle routing problem with time window constraints, Annals of Operations Research, Ann. Oper. Res, Ann Oper R, Ann Oper Res 86 (1999) 585–607.

[19] E. K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Ozcan, J. R. Woodward, A classification of hyper-heuristic approaches.

[20] P. Cowling, G. Kendall, E. Soubeiga, A hyperheuristic approach to scheduling a sales summit, Lecture Notes in Computer Science 2079 LNCS (2001) 176–190.

[21] E. K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. zcan, J. R. Woodward, chap. a classification of hyper-heuristic approaches.

[22] E. K. Burke, B. Mccollum, A. Meisels, S. Petrovic, R. Qu, A graph-based hyper-heuristic for educational timetabling problems.

[23] J. G. Marin-Blazquez, S. Schulenburg, A hyper-heuristic framework with xcs: Learning to create novel problem-solving algorithms constructed from simpler algorithmic ingredients, Lecture Notes in Computer Science) 4399 LNAI (2007) 193–218.

[24] H. Terashima-Marn, E. J. Flores-lvarez, P. Ross, Hyper-heuristics and classifier systems for solving 2d-regular cutting stock problems, Gecco 2005 - Genetic and Evolutionary Computation Conference, Genet. Evol. Comput. Conf (2005) 637–643.

[25] E. Burke, G. Kendall, E. Soubeiga, A tabu-search hyperheuristic for timetabling and rostering, JOURNAL OF HEURISTICS 9 (6) (2003) 451–470. doi:10.1023/b:heur.0000012446.94732.b6.

[26] B. Bilgin, E. zcan, E. E. Korkmaz, An experimental study on hyper-heuristics and exam timetabling.

[27] C. W. Tsai, H. J. Song, M. C. Chiang, A hyper-heuristic clustering algorithm, Conference Proceedings - Ieee International Conference on Systems, Man and Cybernetics, Conf. Proc. Ieee Int. Conf. Syst. Man Cybern (2012) 2839–2844doi:10.1109/icsmc.2012.6378179.

[28] C. Cobos, M. Mendoza, E. Leon, A hyper-heuristic approach to design and tuning heuristic methods for web document clustering, 2011 IEEE CONGRESS ON EVOLUTIONARY COMPUTATION (CEC).

[29] A. C. Kumari, K. Srinivas, M. P. Gupta, Software module clustering using a hyper-heuristic based multi-objective genetic algorithm, PROCEEDINGS OF THE 2013 3RD IEEE INTERNATIONAL ADVANCE COMPUTING CONFERENCE (IACC) (2013) 813–818.

[30] W. Rechenberg, Identification of outliers, FRESENIUS ZEITSCHRIFT FUR ANALYTISCHE CHEMIE 311 (6) (1982) 590–597.

[31] A. Rajaraman, J. Ullman, Mining of Massive Datasets, Cambridge University Press, 2011.
URL http://books.google.dk/books?id=0efRhZyYOb0C

[32] E. Soubeiga, Development and application of hyperheuristics to personnel scheduling, phd (June 2003).
URL http://www.asap.cs.nott.ac.uk/publications/pdf/EricsPhDthesis.pdf

[33] J. Munkres, Algorithms for the assignment and transportation problems, Journal of the Society for Industrial and Applied Mathematics 5 (1) (1957) 32–38. doi:10.1137/0105003.

[34] L. Bodin, B. Golden, A. Assad, M. Ball, Routing and scheduling of vehicles and crews: the state of the art, Computers and Operations Research 10 (2) (1983) 63–211.